A Two-Stage Online Approach for Collaborative Multi-Agent Planning under Uncertainty

Iván Palomares, Kim Bauters, Weiru Liu, and Jun Hong

School of Electronics, Electrical Engineering and Computer Science,
Queen's University Belfast, Belfast, Northern Ireland
{i.palomares,k.bauters,w.liu,j.hong}@qub.ac.uk

Abstract. In a team of multiple agents, the pursuance of a common goal is a defining characteristic. Since agents may have different capabilities, and effects of actions may be uncertain, a common goal can generally only be achieved through a careful cooperation between the different agents. In this work, we propose a novel two-stage planner that combines online planning at both team level and individual level through a subgoal delegation scheme. The proposal brings the advantages of online planning approaches to the multi-agent setting. A number of modifications are made to a classical UCT approximate algorithm to (i) adapt it to the application domains considered, (ii) reduce the branching factor in the underlying search process, and (iii) effectively manage uncertain information of action effects by using information fusion mechanisms. The proposed online multi-agent planner reduces the cost of planning and decreases the temporal cost of reaching a goal, while significantly increasing the chance of success of achieving the common goal.

1 Introduction

Planning is an essential component of autonomous agents. It involves the selection of a series of actions to perform to achieve a goal desired by the agent [19]. Such a series of actions is commonly referred to as *a plan*. Ideally, planning algorithms attempt to take all information about the environment into account when coming up with a plan. However, it is often infeasible to (optimally) plan in realistic environments due to their size and the uncertainty of action outcomes [12]. *Multi-agent planning* is a particular branch of planning where there is a collective approach from multiple agents to achieve a goal [18]. In *collaborative* multi-agent planning, a team of agents try to accomplish a task leading to a common goal by combining their capabilities and knowledge [15]. Two main approaches for collaborative multi-agent planning can be distinguished: (i) *centralised*, which involves a planner agent with full knowledge of the environment and the joint task to undertake, and (ii) *distributed* or decentralised [4], in which agents plan individually and coordinate with each other to find a common solution for the planning problem [10, 13]. Centralised multi-agent planning is typically the most efficient, but is only feasible if agents do not have private or sensitive information [1].

Collaborative multi-agent planning has been an active subject of research in recent years [2, 9, 14, 16]. However, most of these works focus on *offline planning* rather than *online planning*. Online planning differentiates itself from offline planning by not fully

elaborating a plan *before* execution, but instead to interleave planning and execution. To this end, it employs approximate methods such as Monte-Carlo Tree Search (MCTS) [3] to return the next *'good enough'* action rather than a complete series of actions [5]. online planning approaches are their ability to narrow the scope of the search space, to return "good enough" actions anytime and to efficiently re-plan when an unexpected situation is encountered while acting. So far, online planning has been mainly applied to individual agent planning problems, with only a few proposals for online multi-agent planning presented in [11, 20, 21]. Wu et al. [20], use Decentralized POMDPs and stage games for planning in ad-hoc teams, without pre-coordination, such that each agent independently plans its next actions under teamwork considerations. The authors also developed in [21] an online planning approach aimed at minimizing inter-agent communication. Paquet et al. [11] presented a method called Real-Time Belief Space Search (RTBSS) for determining the best next action in large real-time environments.
In this paper we focus on problems and application domains characterised by:

- the existence of a fixed team of agents with common goals requiring coordination;
- planning at team level is required to ensure coordination between agents;
- each agent knows the outcome probabilities of its own actions only.

An example of domains under these settings are SCADA supervisory control systems, e.g. for power grid management, or navigation in hazardous environments such as nuclear sites [6]. The scenario utilised to describe our proposal refers to navigation by multiple robots for clearance in a country park. To the best of our knowledge, problems defined under these settings have not been addressed yet in online planning.

To efficiently solve problems in these domains we introduce a novel two-stage online collaborative planner where actions may have stochastic effects. The first stage is a team level centralised planner which plans on an abstract level and delegates subgoals to individual agents. The second stage is an individual level distributed planner where each agent pursues its assigned subgoal. The proposed planner extends the MCTS-based UCT algorithm [8] to (i) collectively plan for the next best subgoals for every agent in the team, and (ii) to individually come up with suitable plans to achieve the assigned subgoals (individual level planning). A fusion approach [22] is introduced in the team planner to combine uncertain information about the effects of actions, which will help to significantly reduce the search space.
To adequately scope our work we furthermore assume the following principles:

Principle 1 Agents act in parallel to achieve a common goal. A team planner agent determines the next subgoal each agent should individually accomplish.

Principle 2 The agents act in a purely collaborative way, i.e. there is no form of competition in terms of distinct, conflict goals amongst agents. Furthermore, agents carry out their actions independently, such that no effects of interfering the actions of the other agents are considered.

Principle 3 No privacy preservation constraints amongst agents are considered.

The rest of the paper is organised as follows. We start off with some preliminaries in Section 2. In Section 3 the scenario used to illustrate our proposal is introduced.

Our novel, two-stage online multi-agent planner is proposed in Section 4 and evaluated in Section 5, demonstrating its ability to reduce the cost of planning and acting in parallel, as well as increasing the chances of successfully reaching the goal established. Finally, concluding remarks are drawn in Section 6.

2 Preliminaries

In offline planning, a complete plan or course of actions to achieve a goal is firstly generated and then executed by the agent. Therefore, when multiple agents are present and there is no need for preserving private individual information, the planning process can be easily centralised even though execution is performed in a distributed fashion [21]. By contrast, online planning interleaves planning with execution: instead of generating the whole plan a priori, online planners return a next "*good-enough*" action to be executed at the current state. When an unexpected outcome is obtained, online planners can immediately pick up on this new information and do not need to plan in advance for all such eventualities. Our work focuses on integrating online planning at team and individual levels by using online team planning as a delegation scheme.

UCT (*Upper Confidence bounds applied to Trees*) [8] is a state-of-the-art *anytime* algorithm that combines MCTS [3] with multi-bandit selection methods [8], and has been utilised for planning in domains pervaded by uncertainty. UCT [5] allows to quickly return a non-trivial decision after performing a series of *rollouts* in which outcomes of actions are sampled based on their probability. A rollout consists in traversing the search tree from the root node to a node representing a terminal state. Every time a node is visited in UCT, the selection of the action to take at its corresponding state is based on all previous rollouts, favouring actions that either produced higher rewards or were rarely visited in previous rollouts. This allows for a balance between exploitation (selecting actions with better reward statistics so far) and exploration (selecting actions that have still been rarely simulated). A *decision node* in UCT represents an environment state. A decision node corresponding to a non-terminal state can be expanded into available actions at that state, leading to child decision nodes for the outcomes of such actions. The root decision node represents the current environment state [5].

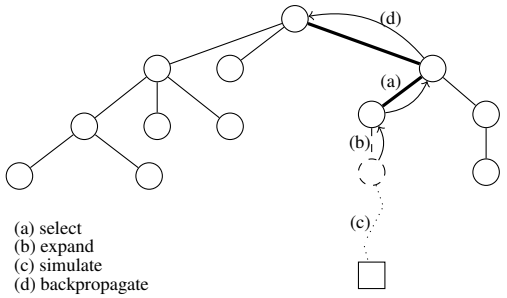

Fig. 1: The four distinct steps in every MCTS iteration.

In every iteration, UCT applies the following four steps (see Figure 1): (a) *Selection*: select a child node based on a selection function. (b) *Expansion*: randomly expand the

selected node to a new unsampled one. (c) *Rollout*: randomly simulate a playout (e.g. a sequence of selected actions and their outcomes) until reaching a terminal state. (d) *Backpropagation*: compute a reward value associated to the terminal state reached, and propagate it back up through the tree to the root node, updating the information for each node in the path.

3 Scenario Overview

The country park scenario serves to illustrate the concepts and ideas presented in this paper. A team of forest management robots (agents) are situated in different locations of a country park, in a region frequently affected by natural disasters such as strong winds and wildfires. After a storm, a number of fallen boulders are detected in locations around the park. The robots, which operate in parallel, must plan and coordinate together to clear the affected locations efficiently. The problem is further complicated by the following factors: (i) the park is organised into a number of locations or *Points of Interest* (PoI) labelled a to n, and a network of hiking trails labelled t_1 to t_{64} connecting the PoIs; (ii) some trails are safer than others due to their width (see Figure 2). Falling off a trail (e.g. into a cliff, due to a landslide, etc.) permanently disables the robot; and (iii) each robot has different competences and/or physical sizes, therefore the probability of successfully crossing a trail can vary from robot to robot. The robots are fully aware of their current position and the position of the boulders in the scenario. Moreover, robots can communicate with the team planner agent to inform about e.g. reaching a new PoI, clearing a boulder, or falling off a trail.

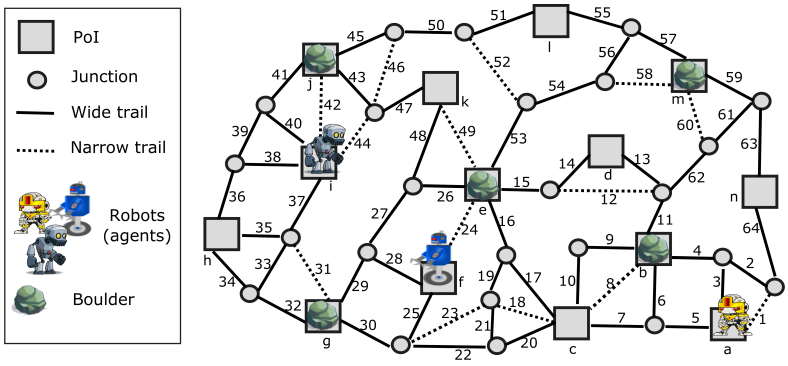

Fig. 2: Country park scenario. PoIs are labeled a to n, and trails (edges) are numbered t_1 to t_{64}

When applying our framework to this scenario the *high-level team planning* will direct robots to neighbouring[1] PoI on their way to reach a boulder to clear. On the *low-level individual planning* the agents themselves will plan for how to reach that neighbouring PoI given their knowledge of the trails and the likelihood of reaching the PoI in any of the available ways given their capabilities/physical sizes.

[1] Neighbouring PoIs are those which can be reached from the current agent position without getting through any other PoI.

4 Online Collaborative Multi-Agent Planner under Uncertainty

In this section we present an online multi-agent planner for collaborative teams of agents whose actions have stochastic outcomes. The main characteristics of the planner are: (1) two planning stages (team planning and individual planning) are interleaved through a subgoal delegation scheme, (2) online planning is utilised through two extensions of the UCT algorithm adapted to both planning phases, and (3) a number of mechanisms are proposed to deal with uncertain stochastic information effectively, whilst preventing an excessive search space.

4.1 Notation and Basic Concepts

The following notation is introduced to refer to the different elements utilised in the proposed multi-agent planner:

- There exists a set $\mathcal{AG} = \{1, 2, \ldots, n\}$ of *agents*.
- There are n *action libraries* $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$, one for each agent $i \in \mathcal{AG}$. An action library $\mathcal{A}_i = \{a_i^1, \ldots, a_i^m\}$ encompasses a finite set of actions a_i^k, $k = 1, \ldots, m$, that can be performed by the agent, i.e. its capabilities. For simplicity, all agents have access to the same actions (e.g. move forward) but the probabilities of outcomes are distinct for each agent.
- A subgoal library $\mathcal{C} = \{c^1, \ldots, c^u\}$ common to all agents describes the possible subgoals that can be assigned to them. A subgoal is achieved by an agent i by applying (a sequence of) actions from its action library $\mathcal{A}_i$, as explained later.

Action, plan and (sub)goal representation is based on PPDDL (Probabilistic Planning Domain Definition Language) [23], as illustrated in several examples throughout this section. PPDDL is fully supported by implementations of MCTS-based techniques.

The set of all possible environment states is represented as $\mathcal{E}$. An *environment state* is denoted by $\epsilon \in \mathcal{E}$, where ϵ_0 denotes the current state, and $\mathcal{E}_G \subset \mathcal{E}$ is the subset of all possible goal states ϵ_G. Since the team planner agent is responsible for the team planning process, it must be able to formulate environment states at team level. A decision node in the search tree constructed during team planning includes these two elements:

1. Collective information about the current state of *all* agents involved in the team planning process, describing each agent's individual status: in our example the positions of robots in the environment.
2. Other purely environmental information: in our example, the locations of remaining boulders, if any.

Thus, a decision node $N(\epsilon)$ associated to an environment state ϵ, is formalised as a 2-tuple $N(\epsilon) = \langle s(\mathcal{AG}); s(env) \rangle$. The set $s(\mathcal{AG}) = \{s_1, \ldots, s_n\}$ denotes the current state of every agent, and $s(env)$ represents environmental information. Conversely, we refer to states modelled in the individual planning phase performed by agent $i \in \mathcal{AG}$ as *agent states*, $\epsilon^i \in \mathcal{E}^i$. Their associated decision nodes $N(\epsilon^i)$ contain more specific information than the (team level) environment states ϵ introduced above, namely information about i and the environment only. They are formally represented as $N(\epsilon^i) =$

$\langle s_i; s(env)\rangle$. In either case (and as occurs with classic UCT), the root decision node describes the current environment (*resp.* agent) state, ϵ_0 (*resp.* ϵ_0^i).

Example 1. Consider the country park scenario (Section 3). Let s_i be the state of agent i. For simplicity, we assume an agent state is solely formed by a predicate of the form, $at(i, L)$, indicating the location L of agent i (which can be either one of the 14 PoIs labeled 'a' to 'n', a junction connecting some of the 64 trails in the park, or the symbol "-" indicating that the agent failed in executing an action and is no longer operating). On the other hand, let $s(env) = \bigwedge_{at(boulder,L)} L$ be the locations of boulders not cleared out yet. A decision node describing this environment state is formalised as follows:

$$N(\epsilon) = \langle \{at(1, a), at(2, f), at(3, i)\}; b \wedge e \wedge g \wedge j \wedge m\rangle$$

with $b \wedge e \wedge g \wedge j \wedge m$ being environmental information (locations of boulders not cleared yet). Suppose that agent 1 plans individually to cross t_5. When reaching the junction connecting t_5, t_6 and t_7 (denoted by $t_{5,6,7}$), its resulting decision node $N(\epsilon^i)$ is:

$$N(\epsilon^1) = \langle at(t_{5,6,7}); b \wedge e \wedge g \wedge j \wedge m\rangle$$

with same environmental information, i.e. no PoIs with boulders have been reached yet. □

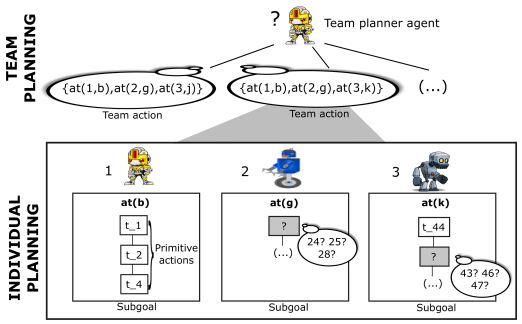

Fig. 3: Actions and subgoals: a team action is associated one or more subgoals, each of which indicates (in our scenario) a target location to be reached e.g. $at(k)$, whereas primitive actions indicate trails to be crossed, e.g. t_44.

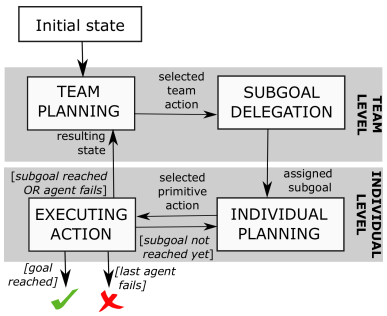

Fig. 4: Multi-agent planner scheme: the team planner delegates subgoals to active agents; each agent in turn plans for achieving its subgoal and relegates execution results back to the team planner.

We now introduce the three central concepts in the proposed planner: *primitive action*, *subgoal* and *team action*. These concepts are illustrated in Figure 3 to facilitate their understanding.

Definition 1. *A primitive action $a_i^k \in \mathcal{A}_i$ can be individually undertaken by agent $i \in \mathcal{AG}$, e.g. t_1 for the action of moving across trail '1'. Primitive actions are evaluated and selected during individual planning.*

Definition 2. *A subgoal $c_i \in \mathcal{C}$ assigned to agent i represents an individual state i should aim for, e.g. $at(a)$, which indicates that i must reach the PoI labeled 'a'. Subgoals are assigned by the team planner agent to every agent in the team.*

Definition 3. *A team action $\tau = \{\sigma_i, i \in pa(\tau)\}$ encompasses a number of subgoals σ_i simultaneously assigned to a team of participating agents $pa(\tau) \subseteq \mathcal{AG}$ (one subgoal per agent) at a given time. Team actions are formulated during team planning, and they involve those agents that need a new subgoal to be pursued at a given state.*

Example 2. Suppose that the following team action is selected as a result of team planning in the country park scenario, $\tau = \{at(1, n), at(2, e), at(3, g)\}$. This means that the subgoal of reaching location 'n', $at(n)$, is delegated to robot 1, the subgoal $at(e)$ is delegated into 2 and the subgoal $at(g)$ is delegated into 3. □

Outcomes of team actions are environment states that result either from the successful accomplishment by every agent of its assigned subgoal, or from one or more agents in $pa(\tau)$ failing to accomplish it (being deemed unavailable hereinafter because e.g. they fell off a trail). In the search tree constructed during team planning, decision nodes (except for $N(\epsilon_0)$) represent outcomes of team actions.

A general scheme of the proposed two-stage multi-agent planner is depicted in Figure 4 and explained in the following two subsections.

4.2 Team Planning

The team planning process aims at determining the next best team action τ describing the immediate subgoals each agent is assigned. This is not a straightforward task for two reasons: (1) determining the probabilities of occurrence of each available team action requires stochastic information about individual action libraries, and (2) having multiple agents acting in parallel may involve a significant number of possible outcomes for τ.

To cope with these difficulties, we firstly distinguish two distinct types of outcomes for any τ. A *success outcome* occurs when *all* agents in $pa(\tau)$ succeed in achieving their respective subgoals. A special case of a success outcome is when the common goal has been achieved. In such a case the success outcome is also a *goal state*. Whenever we do not have a success outcome, we say that we have an *undesired outcome*. Both goal states and undesired outcomes are regarded as *terminal states*[2].

Based on this distinction, we can now focus on defining a reward-driven online team planner. In particular, we will introduce a method based on uncertain information fusion to estimate rewards of team actions. Next, we describe how the phases of the UCT algorithm are adapted to deal with such team actions. The subsequent individual planning phase (Section 4.3) describes how each agent accurately plans to pursue its assigned subgoal, taking account of its probabilities of action outcomes.

Reward Estimation at Team Level. In UCT, a reward function assigns a value to the terminal state encountered at the end of a rollout before it is backpropagated: the greater this value, the more rewarding the outcome is deemed. Below we introduce a collective reward function that allows to preserve a reduced branching factor in the search tree by summarising all possible forms of undesired outcome into one. This general function must be instantiated to suit the specific scenario tackled.

[2] Undesired outcomes are considered as terminal states: if an unexpected situation is encountered, the remaining agents start another planning process upon the resulting environment state.

Definition 4. *Let $\mathcal{E}_F(\subset \mathcal{E})$ be a set of all undesired outcomes and $\mathcal{E}_G$ a set of all goal states, as defined in Section 4.1. The set of all terminal states is given by $\mathcal{E}_G \cup \mathcal{E}_F = \mathcal{E}_T$, and $\mathcal{E}_{\overline{\tau}} \subset \mathcal{E}_F$ is the set of all the possible undesired outcomes ϵ_F of τ. We propose summarising such outcomes as one in the search tree, therefore $\mathcal{E}_{\overline{\tau}}$ is deemed as a single terminal state hereinafter for the reward computation of undesired outcomes. We define a reward function f as a mapping $f : \mathcal{E}_T \rightarrow [-1, 1]\backslash\{0\}$, with the following properties:*

(i) $f(\epsilon_G) > 0$, $\forall \epsilon_G \in \mathcal{E}_G$, *i.e. arriving at a goal state always produces a positive reward value.*

(ii) $f(\mathcal{E}_{\overline{\tau}}) < 0$, $\forall \mathcal{E}_{\overline{\tau}} \subset \mathcal{E}_F$, *i.e. arriving at any undesired outcome always produces a negative reward value.*

(iii) Let $d \in \mathbb{N}$ be the depth level at which the terminal state is encountered. Assume two identical terminal states ϵ_1, ϵ_2 can be reached at depth d_1 and d_2 respectively, with $d_1 < d_2$. Then $f(\epsilon_1) \geq f(\epsilon_2)$.

The computation of $f(\mathcal{E}_{\overline{\tau}}) < 0$ (property (ii)) is based on the aggregation of information related to each form of undesired outcome $\epsilon_F \in \mathcal{E}_{\overline{\tau}}$, as explained below. It follows from property (iii) that similar goal states lead to an equal or higher reward when they are encountered after a lower number of consecutive team actions. Similarly, undesired outcomes are equally or less detrimental when encountered earlier. A discount factor $\delta \in]0, 1[$ can be applied in f to reflect this property.

The reward value for an undesired outcome of τ is defined as follows. Based on each $\epsilon_F \in \mathcal{E}_{\overline{\tau}}$, two indicators $\varphi(\epsilon_F), \gamma(\epsilon_F) \in [0, 1]$ are introduced for *resp.* (i) the number of agents in the team who fail to accomplish their subgoal σ_i in τ, $|fa(\tau)|$, with respect to the total number of participating agents; and (ii) the resulting "distance" to the (closest) goal state. The former is computed as $\varphi(\epsilon_F) = |fa(\tau)|/|pa(\tau)|$, whereas the latter is domain-dependent. For our scenario, it is calculated based on the number of remaining boulders when ϵ_F occurs, i.e. $\gamma(\epsilon_F) = \#remaining/\#boulders$.

In addition, $|\mathcal{E}_{\overline{\tau}}|$ is the total number of possible undesired outcomes ϵ_F of τ. This parameter is calculated in our scenario as $|\mathcal{E}_{\overline{\tau}}| = 2^{|pa(\tau)|} - 1$, because the number of possible outcomes only depends on the (possible subsets of) agents in $pa(\tau)$ which fail in achieving their assigned subgoal. Hence, $f(\mathcal{E}_{\overline{\tau}})$ is defined as follows:

$$f(\mathcal{E}_{\overline{\tau}}) = -\delta^{d-1} \frac{\sum_{\epsilon_F \in \mathcal{E}_{\overline{\tau}}} \mathcal{U}\left(\varphi(\epsilon_F), \gamma(\epsilon_F)\right)}{|\mathcal{E}_{\overline{\tau}}|} \tag{1}$$

with $\mathcal{U} : [0, 1]^2 \rightarrow [0, 1]$ a uninorm aggregation function [22], that combines the two indicators φ, γ into a single value (as explained below). The fusion procedure applied in Equation (1) for reward computation eliminates the need for splitting undesired outcomes into multiple leaf nodes. This significantly simplifies the search tree constructed.

Uninorm aggregation functions are a generalisation of t-norm and t-conorm functions [22] with a neutral element $g \in]0, 1[$, fulfilling the full reinforcement property, i.e. if the two values to aggregate $x, y \in [0, 1]$ are both higher (*resp.* lower) than g, then the aggregated result becomes even higher (*resp.* lower). Conversely, they present a compensating (averaging) behaviour if one of the values is high and the other is low. The reinforcement property is particularly interesting in the application domains considered in this paper to emphasise situations when:

(i) There are few remaining agents, far away from reaching their goal, in which case both φ and γ are high and the aggregated value is reinforced upwards.

(ii) Most agents still remain and they are close to the goal, in which case φ, γ are low and the aggregated value is reinforced downwards.

The use of uninorm functions affects therefore the assessment of single undesired outcomes $\epsilon_F \in \mathcal{E}_{\overline{\tau}}$ in the two cases outlined above. Because of the minus sign in Equation (1), in our context $\mathcal{U}$ behaves as a cost function: the higher its value for a given outcome $\epsilon_F \in \mathcal{E}_{\overline{\tau}}$, the less rewarding this outcome is, hence the lower the resulting $f(\mathcal{E}_{\overline{\tau}})$ will be. An example of these functions is the cross-ratio uninorm with $g=0.5$ [7]:

$$\mathcal{U}(x,y) = \begin{cases} 0 & (x,y) \in \{(0,1),(1,0)\}, \\ \dfrac{xy}{xy+(1-x)(1-y)} & \text{otherwise.} \end{cases} \tag{2}$$

Regarding reward computation for goal states, since we consider problems where all agents share a common goal, the reward function for any $\epsilon_G \in \mathcal{E}_G$ can be simply defined as $f(\epsilon_G) = \delta^{d-1}$, i.e. the sooner the goal is accomplished (lower d), the less resources are consumed by agents to reach it, hence the more beneficial the outcome is.

Example 3. Assume the current state of the environment in the country park scenario is given by $N(\epsilon_0)=\langle\{at(1,n), at(2,k), at(3,-)\};j\wedge m\rangle$, which means that robots 1 and 2 are active and situated in 'n' and 'k' respectively, whereas 3 already fell off a trail, and the only remaining boulders are located in 'j' and 'm'. One of the available team actions for $pa(\tau) = \{1,2\}$ is $\tau = \{at(1,m), at(2,j)\}$, whose completion intuitively implies achieving the overall team goal, in which case $d = 1$ and $f(\epsilon_G) = \delta^{d-1} = 1, \forall\delta$. The reward of reaching the undesired outcome is computed based on Eqs. (1) and (2):

$$f(\mathcal{E}_{\overline{\tau}}) = -\frac{\mathcal{U}(0.5,0.33)+\mathcal{U}(0.5,0.33)+\mathcal{U}(1,0.66)}{3} = -0.55$$

□

UCT-based Search Process at Team Level. Assuming that a team action can either lead to a success outcome, or to a(n) (summarised) undesired outcome, the collective search tree structure is adapted as follows: every edge representing a team action leads to a *node pair* formed by the decision nodes associated to the success outcome and the undesired outcome (see Figure 5). The latter is regarded as a leaf node (terminal state), as explained earlier. If, however, the success outcome of the node pair does not represent a goal state, it can be expanded into a number of next available team actions at that state.

The backpropagation process is now adapted to the proposed node pair structure. We firstly explain how rewards are updated through nodes generated during rollout, up to the last expanded node. Rewards throughout the rollout path cannot be accurately calculated, since a team action τ_j immediately taken at a non-terminal state can eventually lead to different terminal states with varying rewards. Notwithstanding, it is possible to estimate the *"best and worst possible scenario"* that might be encountered at any state

in the rollout path. In other words, given a state ϵ in the rollout path, we can estimate the highest (*resp.* lowest) reward that could be eventually achieved after applying a number of team actions posterior to ϵ. Based on this, we propose modeling the reward of a non-terminal state $\epsilon_S \in \mathcal{E} \backslash \mathcal{E}_T$ as an interval, $f(\epsilon_S) = f(\tau_j) = [f(\tau_j)^-, f(\tau_j)^+]$, with $f(\tau_j)^- \in [-1, 0[$ and $f(\tau_j)^+ \in]0, 1]$. Here, τ_j is the (unique) team action generated upon ϵ_S during rollout, therefore the interval-valued reward is easily calculated as:

$$f(\tau_j) = [\min_{\tau_k \dashv \tau_j} f(\mathcal{E}_{\overline{\tau}_k}), f(\epsilon_G)] \tag{3}$$

where $\tau_k \dashv \tau_j$ represents all rollout actions τ_k posterior to τ_j. The shaded area in Figure 6 illustrates backpropagation through rollout nodes up to $N(\epsilon_1)$.

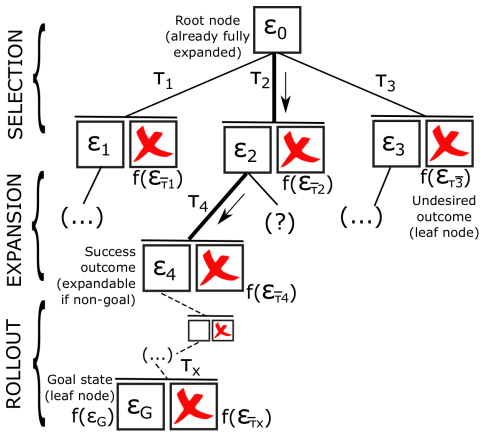

Fig. 5: UCT-based search in team planning

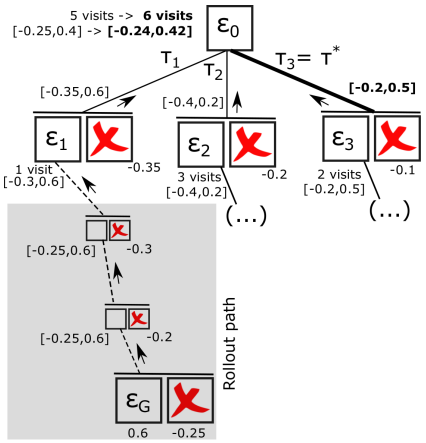

Fig. 6: Team planner backpropagation

Backpropagation between the last expanded node and the root node updates rewards and also increases the visit count for nodes in the path. However, given a node $N(\epsilon'_S)$ resulting from applying τ_j at a previous state ϵ_S, the reward interval backpropagated to $N(\epsilon_S)$ is not necessarily $f(\tau_j)$, but instead that of the most rewarding action available at ϵ_S. It is therefore necessary to compare the interval-valued rewards of all available actions at ϵ_S and backpropagate the highest one. To do this, the method in [17] to calculate the preference degree between intervals of real numbers is utilised:

$$P(\tau_j > \tau_k) = \frac{\max(0, f_j^+ - f_k^-) - \max(0, f_j^- - f_k^+)}{(f_j^+ - f_j^-) + (f_k^+ - f_k^-)} \tag{4}$$

where interval bounds $f(\tau_j)^-$ are denoted as f_j^- for simplicity. This allows to determine the most rewarding available action τ^* at ϵ_S. Rewards $f(\epsilon_S) = [f(\epsilon_S)^-, f(\epsilon_S)^+]$ are then updated based on the number of visits its corresponding node received so far:

$$f(\epsilon_S)^- = \frac{f(\tau^*)^- + \#visits \cdot f(\epsilon_S)^-_{old}}{\#visits + 1} \quad f(\epsilon_S)^+ = \frac{f(\tau^*)^+ + \#visits \cdot f(\epsilon_S)^+_{old}}{\#visits + 1} \tag{5}$$

In Figure 6, τ_3 is more rewarding than the team action in the backpropagation path, τ_1. Therefore, rewards in $N(\epsilon_0)$ are updated based on $f(\tau_3) = [-0.2, 0.5]$.

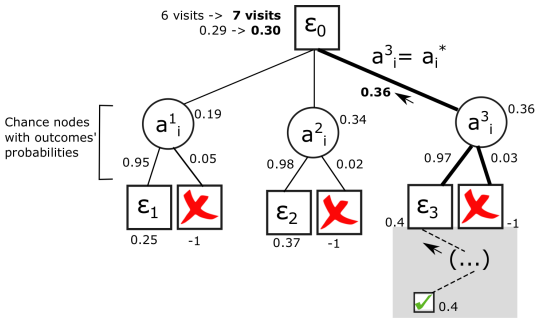

Fig. 7: Individual search tree structure and rollout-backpropagation after expanding into a_i^3

After a number of iterations, the *best* next team action $\tau = \{\sigma_i, i \in pa(\tau)\}$ is returned. The team planner then delegates the subgoal σ_i into each participating agent i, which proceeds to the individual planning phase to pursue the assigned subgoal.

4.3 Individual Planning

The online approach utilised for the individual planning phase is a standard UCT-based approach with multiple reward rollouts at each iteration. We distinguish two types of nodes between which the algorithm alternates during construction of the tree: decision nodes and chance nodes. The latter represent available primitive actions at the state described by their parent decision node. Each chance node has in turn a number of children decision nodes, one for each possible action outcome. The tree structure is represented in Figure 7.

When a decision node $N(\epsilon^i)$ is expanded, a child chance node representing one of the available actions at that state is generated. New decision nodes for the outcomes of the newly generated chance node are also added to the tree. The subsequent rollout phase of UCT is modified so that, at each iteration of the algorithm, a number r of rollouts are carried out for each non-terminal outcome[3]. This allows to quickly obtain accurate reward estimates for the state from which rollouts are being currently performed, as well as thoroughly exploring the different courses of action available from each outcome. Each rollout takes place until a terminal state (either subgoal achievement or failure) is encountered, and it is followed by the backpropagation and cumulation of the reward obtained up to the root node. The reward value of every individual fail state ϵ_F^i is instantiated as $f(\epsilon_F^i) = -1$, as failing any primitive action in our scenario implies that the agent is no longer available. On the other hand, for an individual (sub)goal state ϵ_G we again have $f(\epsilon_G^i) = \delta^{d-1}$. The reward update process between the last expanded node and the root node is applied differently for each type of node (see Figure 7):

1. *Chance node*: The reward $f(a_i^k) \in [-1, 1]$ of a chance node associated to a_i^k is calculated as the probability-weighted mean of its outcomes' rewards.

[3] In the country park scenario, primitive actions have at most one non-terminal outcome, but this could not be the case in other different scenarios with multiple stochastic action outcomes.

2. *Decision node*: Rewards of decision nodes are updated similarly as explained in the team planner, with the only difference that individual rewards of non-terminal states are real values instead of intervals. Assuming that a_i^* is the most rewarding available action at ϵ^i, the reward in $N(\epsilon^i)$ is updated as follows:

$$f(\epsilon^i) = \frac{f(a_i^*) + \#visits \cdot f(\epsilon^i)_{old}}{\#visits + 1} \tag{6}$$

5 Experiments and Results

In this section we demonstrate the performance of the proposed multi-agent planner. Throughout the evaluation, we refer to the country park scenario from Section 3, and the problem formulation shown in Figure 2. To evaluate the performance of our proposed *two-stage* multi-agent planner, we consider the following two baselines:

1. *one-stage multi-agent planner*: this baseline coincides with a fully centralised planner, which controls the actions of each individual agent. We implemented this baseline as a simplification of our proposed planning framework, where the team planner directly plans over primitive actions of agents. Team actions are thus composed of primitive actions instead of subgoals.
2. *multiple agents planning individually*: each agent plans independently and individually for the primitive actions to achieve the overall goal of clearing *all* boulders from PoIs. In this baseline there is no coordination schema amongst agents. To make the baseline more goal-aware, agents do communicate with each other to update their environmental information when necessary, e.g. if a PoI has been cleared.

In the experiment we pit our novel two-stage multi-agent planner against both baselines as discussed above. Each approach is used to solve 100 instances of the park scenario (see Figure 2). The following metrics are subsequently considered:

(i) $\%success$: percentage of executions in which the goal is achieved (higher is better);

(ii) $\#actions$: total number of primitive actions undertaken by all agents per execution, before achieving the goal or failing to complete it (lower is better).

The first metric gives an indication of how good each approach is in tackling this particular scenario, whereas the second metric gives an indication of the temporal complexity of the solutions found by each method.

Table 1: Comparison of success rate and average number of primitive actions

	two-stage	one-stage	individually
% success	85	62	52
avg. # actions	15.96	21.68	34.63

Table 1 summarises the resulting values of each metric for the three planning approaches being compared. Figure 8 depicts the value of $\#actions$ obtained by the proposed planner for each execution, compared to those obtained by the two baseline planning approaches. Our results show that a team of agents coordinated by the proposed two-stage framework and acting in parallel outperform both baseline approaches,

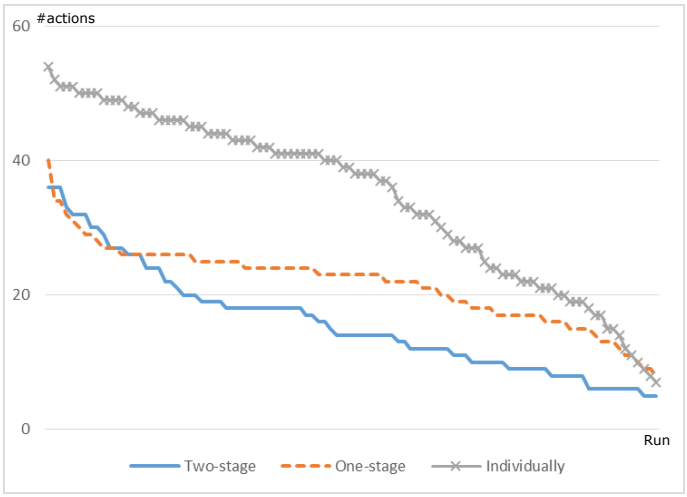

Fig. 8: Comparison of the number of primitive actions (ordered from most to least) undertaken by agents per execution

in terms of the temporal cost (number of required primitive actions) to reach the goal (particularly compared to the individual planning baseline); along with a significant increase in the planning robustness, i.e. the chances of successfully reaching the common goal. Based on these results, we conclude that our two-stage online multi-agent planning approach endowed with a subgoal delegation mechanism allows for higher robustness and lower cost in the planning domains under uncertainty considered. Furthermore, in the scenario considered, the inclusion of a subgoal delegation scheme intuitively allows for a significant reduction in search space, compared to planning at team level over primitive actions directly.

6 Conclusions

In this paper, we have presented a two-stage online collaborative multi-agent planner for application domains where agent actions have uncertain stochastic effects. The proposed planner interleaves team and individual online planning through a subgoal delegation scheme, and extends state-of-the-art approximate algorithms to suit the characteristics of the planning problems considered. The proposed framework estimates rewards of action outcomes at team level, by using uncertain information fusion procedures, to determine the next best subgoals to be individually pursed by each agent in the team. Future lines of investigation aim at developing data-driven online team planning approaches that enable precise estimations of outcome probabilities of team actions alongside rewards, and the integration of prunning policies in both planning stages.

Acknowledgments

This work has been funded by EPSRC PACES project (Ref: EP/J012149/1).

References

1. Brafman, R., Domshlak, C.: From one to many: Planning for loosely coupled multi-agent systems. Proc. of ICAPS'08 (2008)
2. Brafman, R.I.: A privacy preserving algorithm for multi-agent planning and search. In: Proc. of IJCAI'15. pp. 1530–1536. IJCAI'15 (2015)
3. Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A Survey of Monte Carlo Tree Search Methods. IEEE Transactions on Computational Intelligence and AI in Games 4(1), 1–43 (March 2012)
4. Durfee, E.: Distributed problem solving and planning. In G. Weiss (Ed.): A Modern Approach to Distributed Artificial Intelligence (1999)
5. Keller, T., Eyerich, P.: PROST: Probabilistic planning based on UCT. In: Proc. of ICAPS'12 (2012)
6. Killough, R., Bauters, K., McAreavey, K., Liu, W., Hong, J.: Risk-aware planning in BDI agents. In: Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART'16) (2016)
7. Klement, E., Mesiar, R., Pap, E.: On the relationship of associative compensatory operators to triangular norms and conorms. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 04(02), 129–144 (1996)
8. Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In: Proc. of ECML'06. pp. 282–293. Springer-Verlag, Berlin, Heidelberg (2006)
9. Marcolino, L.S., Matsubara, H.: Multi-agent monte carlo Go. In: Proc. of AAMAS'11. pp. 21–28. International Foundation for Autonomous Agents and Multiagent Systems (2011)
10. Melo, F.S., Sardinha, A.: Ad hoc teamwork by learning teammates' task. Autonomous Agents and Multi-Agent Systems 30(2), 175–219 (2015)
11. Paquet, S., Tobin, L., Chaib-draa, B.: An online POMDP algorithm for complex multiagent environments. In: Proc. of AAMAS '05. pp. 970–977 (2005)
12. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson Education (2009)
13. Semsar-Kazerooni, E., Khorasani, K.: Multi-agent team cooperation: A game theory approach. Automatica 45(10), 2205 – 2213 (2009)
14. Smith, R.: Coordination of temporal plans in dynamic environments for mobile agents. Laboratoire d'Informatique de Paris VI (2012)
15. Torreño, A., Onaindia, E., Sapena, O.: FMAP: Distributed cooperative multi-agent planning. Applied Intelligence 41(2), 606–626 (2014)
16. Torreño, A., Sapena, O., Onaindia, E.: Global heuristics for distributed cooperative multi-agent planning. In: ICAPS 2015. 25th International Conference on Automated Planning and Scheduling. pp. 225–233. AAAI Press (2015)
17. Wang, Y., Yang, J., Xu, D.: A preference aggregation method through the estimation of utility intervals. Computers and Operations Research 32, 2027–2049 (2005)
18. de Weerdt, M., Clement, B.: Introduction to planning in multiagent systems. Multiagent Grid Syst. 5(4), 345–355 (2009)
19. Weld, D.: Recent advances in AI planning. AI Magazine 20, 93–123 (1999)
20. Wu, F., Zilberstein, S., Chen, X.: Online planning for ad hoc autonomous agent teams. In: Proc. of IJCAI'11. pp. 439–445 (2011)
21. Wu, F., Zilberstein, S., Chen, X.: Online planning for multi-agent systems with bounded communication. Artificial Intelligence 175(2), 487 – 511 (2011)
22. Yager, R., Rybalov, A.: Uninorm aggregation operators. Fuzzy Sets and Systems 80, 111–120 (1996)
23. Younes, H., Littman, M.: PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. In: Proc. of ICAPS'03 (2003)